John von Neumann Institute for Computing

Auto-Optimization of Linear Algebra Parallel Routines: The Cholesky Factorization

L.-P. García, J. Cuenca, D. Giménez

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 229-236, 2006.

http://www.fz-juelich.de/nic-series/volume33

Auto-optimization of linear algebra parallel routines: the Cholesky factorization

Luis-Pedro García[a], Javier Cuenca[b], Domingo Giménez[c]

[a]Servicio de Apoyo a la Investigación Tecnológica, Universidad Politécnica de Cartagena, 30203 Cartagena, Spain

[b]Departamento de Ingeniería y Tecnología de Computadores, Universidad de Murcia, 30071 Murcia, Spain

[c]Departamento de Informática y Sistemas, Universidad de Murcia, 30071 Murcia, Spain

1. Introduction

In recent years different techniques to obtain software able to tune automatically to the conditions of parallel platforms have been studied [3,9,10,12,16]. Such developments will facilitate efficient utilization of the routines by non-expert users, e.g. those normally using linear algebra routines in the solution of large scientific or engineering problems.

One approach to obtain self-optimized linear algebra routines is the modelling of the behavior of the algorithm [5–8]. In this paper, this technique is applied to a parallel routine for the Cholesky factorization in message-passing systems. In order to obtain a good estimation with the model, the use of different costs for different types of MPI communication mechanisms (user-defined datatypes or predefined datatypes) is analyzed. The algorithm is studied in systems in which it is possible to use more than one interconnection network simultaneously. In addition, in platforms with a high ratio of the communication cost with respect to the computation cost, it is convenient to take into account that the cost of the communication parameters can vary with the volume of the communication.

The parallel Cholesky factorization is obtained with a block-cyclic partitioning in a logical two-dimensional mesh of $p = r \times c$ processes (in ScaLAPACK [4] style). An analytical model of the execution time of the parallel algorithm is developed as a function of the problem size, the system parameters (parameters of the target platform) and the algorithmic parameters. The algorithm is studied both theoretically and experimentally in order to determine the effect of the value of the system parameters on the selection of the algorithmic parameters. The typical system parameters considered in the study are the cost of arithmetic operations using BLAS kernels of levels 1, 2 or 3 (k_1, k_2, k_3) and the cost of the communication parameters (start-up, t_s, and word sending time, t_w) for the MPI library used. The algorithmic parameters are the block size, b (a block based algorithm is considered), and the parameters r and c defining the logical topology of the processes grid.

Along with the analytical model for the routine, an information system is designed and joined to the routine, providing auto-optimization capacity. The life-cycle of the auto-optimized routine can be summarized as follows [7]:

- In the design phase, the designer creates the routine if a new one is being designed. The complexity of the routine is studied, obtaining an analytical model of its execution time. Next, the installation engine and the optimization manager are created.
- In the installation phase, the system manager instals the routine, guided by the previously designed installation engine and optimization manager, and the information about each system parameter is obtained.

- In the execution phase, the optimization manager obtains information about the current system state. Next the system parameters are tuned according to the system state, and with the analytical model the optimum algorithmic parameters are selected. Finally, the routine is executed.

The remainder of the paper is structured as follows. In Section 2, the parallel Cholesky factorization is analyzed and the system parameters, the algorithmic parameters and the model for the execution time are obtained. In Section 3, experimental results are shown. Section 4 summarizes the work.

2. Parallel routine by blocks for the Cholesky factorization

In a previous work [5], a sequential block Cholesky factorization was analyzed. In that case, the block size (b) and the best library were selected to obtain execution times close to the optimum. Now a parallel implementation is considered, the $n \times n$ matrix is mapped with a block cyclic 2-D distribution onto a two-dimensional mesh of $p = r \times c$ processes.

The Cholesky factorization routine involves the following operations (figure 1 (a)):

- Process $\{0,0\}$ computes the factor L_{11} (lower triangular Cholesky factor of A_{11}).
- The processes in column 0 of the mesh compute $L_{21} = A_{21}(L_{11}^T)^{-1}$.
- All processes participate in the updating $\tilde{A}_{22} = A_{22} - L_{21}L_{21}^T = L_{22}L_{22}^T$.

In figures 1 (b) and 1 (c) the distribution of the work in the following steps is shown.

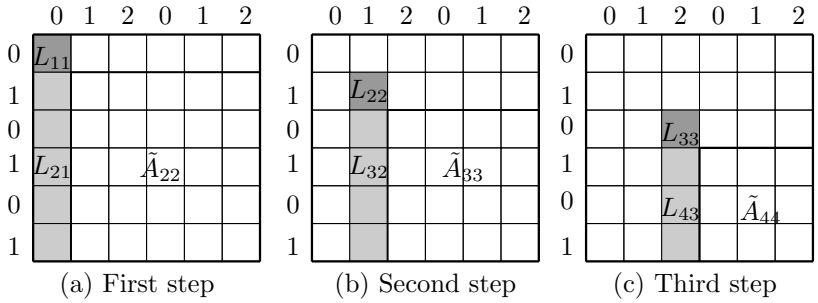

Figure 1. Work distribution in the three first steps of the parallel Cholesky routine by blocks, with $\frac{n}{b} = 6$ and $p = 2 \times 3$. The numbers on the left and on the top of the matrix represent coordinates in the 2×3 processes mesh.

The different parts of the Cholesky routine are identified in order to build the analytical model of the execution time:

- **PPOTF2** The process $\{r_i, c_j\}$, which has the $b \times b$ diagonal block A_{11}, performs the Cholesky factorization of A_{11} and computes L_{11} using the level 2 LAPACK **dpotf2** routine.
- **PTRSM** The process $\{r_i, c_j\}$ broadcasts L_{11} along the column of processes, and all the processes in the column compute the column of blocks of L_{21} by solving a triangular system of size $(n - ib) \times b$ using the level 3 BLAS **dtrsm** routine.

- **PSYRK** The column of processes with the blocks of L_{21} broadcasts columnwise their local part to the other processes in their same column, then L_{21} is broadcast rowwise. Now, all processes can update their local part of A_{22} ($\tilde{A}_{22} \leftarrow A_{22} - L_{21}L_{21}^T = L_{22}L_{22}^T$). This operation is made using BLAS routines of level 3: **dsyrk** (diagonal blocks) or **dgemm** (non diagonal blocks).

The following assumptions for the model of communications in the parallel computer are made. The parallel computer comprises a number of nodes. Each node comprises one or several identical processors interconnected by a switched communication network. The time taken to send a message of size n between any two nodes is independent of the distance between nodes and can be modelled as $t_{comm}(n,p) = t_s(p) + nt_w(n,p)$, where t_s is the latency or start-up time of the message, and t_w is the transfer time per data. The links between two nodes are full-duplex and single ported: a message can be transferred in both directions by the link at the same time, and only one message can be sent and one message can be received at the same time. Because more than a communication network can be available in the parallel computer, the values of t_s and t_w can be different between different processes, and we will have $t_{comm}(n,p)_{i,j} = t_s(p)_{i,j} + nt_w(n,p)_{i,j}$.

For this algorithm, the system parameters are: for the arithmetic cost in LAPACK and BLAS3 routines the computation cost of operations of level 2 and 3 (k_2 and k_3), and for the communication cost of the MPI library, t_s and t_w. In the sequential algorithm the only algorithmic parameter is the block size b, but in the parallel case the number of processes, p, and the dimensions of the logical two-dimensional mesh ($p = r \times c$) are additional algorithmic parameters. Thus, the values of the system parameters will depend on those of the algorithmic parameters, the problem size (n) and the library used (l). Since different level 2 and 3 routines are used, the cost with each one of them can be different and different parameters are considered: $k_{2,potf2}$, $k_{3,trsm}$, $k_{3,gemm}$ and $k_{3,syrk}$. If the algorithm uses different types of communications, the cost of the communication parameters varies. In order to show this variation, different types of broadcast communications are considered. The communication of blocks between processes on the same column is performed with MPI derived data types [14] with a cost $t_s + b^2 t_{w_d}$, and the communication of columns of blocks between processes rows is performed with MPI predefined data types, with cost $t_s + b^2 t_{w_s}$.

Thus, the execution time can be modelled by the formulas:

$$\begin{aligned} t_{arit} = {} & k_{2,dpotf2}\frac{nb^2}{3} + k_{3,dtrsm}\left[\frac{n}{r}(r-1) + \frac{n}{2}\left(\frac{n}{rb}-1\right)\right]b^2 \\ & + k_{3,dsyrk}\left\lceil\frac{1}{\sqrt{p}}\right\rceil\left(\frac{n^2-nb}{2}\right)(b+1) + \frac{2}{p}k_{3,dgemm}\left(\frac{n^3}{6} - \frac{n^2 b}{2} + \frac{nb^2}{3}\right) \end{aligned} \tag{1}$$

for the arithmetic cost, and:

$$t_{com} = \left(\frac{n}{b}-1\right)\left(t_s + b^2 t_{w_d}\right) + \frac{n}{2b}\left(\frac{n}{b}-1\right)\left(t_s + b^2 t_{w_d}\right) + \left(\frac{n}{b}-1\right)\left(bt_s + \frac{nb}{2}t_{w_s}\right) \tag{2}$$

for the communication cost.

3. Experimental Results

Experiments have been performed on two different systems in order to study the analytical model developed:

- A network of four nodes Intel Pentium 4 (P4net) with a switch FastEthernet, enabling parallel communications between them. The MPI library used is MPICH [2].
- A network of four nodes HP AlphaServer quad processors (HPC160) using Shared Memory (HPC160smp), MemoryChannel [11] (HPC160mc) or both (HPC160smp-mc) for the communication between processes. A MPI library optimized for Shared Memory and for MemoryChannel has been used [1].

The values of the arithmetic and the communication parameters are estimated with routines performing some basic operations with the same data access scheme used in the algorithm and with routines that communicate processes in the logical mesh. In both cases, the experiments were repeated several times to obtain an average value.

In an algorithm by blocks, the block size and the library to be used must be decided. Since with the optimized version of BLAS and LAPACK the lowest execution times are obtained, only these libraries are used in the experiments whose results are shown. Therefore, the arithmetic system parameters and their dependency with respect to the algorithmic parameters are shown with BLAS optimized for Pentium 4 (BLAS4) [15] and for Alpha (CXML) [13]. The values of $k_{2,potf2}$ can be considered constants with respect to r and c, but they depend on n and b. The other arithmetic system parameters, $k_{3,dgemm}$, $k_{3,dsyrk}$ and $k_{3,dtrsm}$, can be considered as a function of only the block size. In tables 1 and 2 the values used in the model are shown.

Block size	32	64	128	256
$k_{3,dgemm}$	0,001862	0,000937	0,000572	0,000467
$k_{3,dsyrk}$	0,003492	0,001484	0,001228	0,000762
$k_{3,dtrsm}$	0,011719	0,006527	0,003785	0,002325

Table 1
Values of $k_{3,dgemm}$, $k_{3,dsyrk}$ and $k_{3,dtrsm}$ (in μsec) for different block sizes, in Pentium 4 with BLAS4.

Block size	32	64	128	256
$k_{3,dgemm}$	0,000824	0,000658	0,000610	0,000580
$k_{3,dsyrk}$	0,001628	0,001164	0,000807	0,000688
$k_{3,dtrsm}$	0,001617	0,001110	0,000841	0,000706

Table 2
Values of $k_{3,dgemm}$, $k_{3,dsyrk}$ and $k_{3,dtrsm}$ (in μsec) for different block sizes, in HPC160 with CXML.

In P4net the interconnection network is very slow in comparison with the speed of the processors. It is necessary to take into account that the word sending time, t_w, varies with the message size. Table 3 shows the values obtained experimentally for the cost of the transfer time with MPI predefined data types, t_{w_s}, between processes in a same row; and table 4 for MPI derived data type, t_{w_d}, between processes in the same column. The values of the start-up time, t_s, can be considered as a function of only the number of processes, and can be approximated by $t_s = 55\ \mu sec$ for $p = 2$ and $t_s = 121\ \mu sec$ for $p = 4$.

	Message size		
p	1500	2048	> 4000
2	0,61	0,77	0,84
4	1,22	1,45	1,68

Table 3
Values of t_{w_s} (in μsec) obtained experimentally for different message sizes and number of processes. In P4net.

In HPC160mc and HPC160smp with faster interconnection networks, t_{w_s} can be considered as a function of only the number of processes, and the measured values are approximately $t_{w_s} = 0,011$ μsec for $p = 2$ and $t_{w_s} = 0,025$ μsec for $p = 4$ in HPC160smp, and $t_{w_s} = 0,072$ μsec for $p = 2$ and $t_{w_s} = 0,14$ μsec for $p = 4$ in HPC160mc. But with derived data types it is necessary to consider that the word sending time, t_{w_d}, varies with the block size (table 4). The values of the start-up time t_s can be considered as a function of only the number of processes, and can be approximated by $t_s = 4,88$ μsec for $p = 2$ and $t_s = 9,77$ μsec for $p = 4$, in HPC160smp and in HPC160mc.

	Block size											
	P4net				HPC160smp				HPC160mc			
p	32	64	128	256	32	64	128	256	32	64	128	256
2	0,97	0,84	1,00	1,10	0,019	0,024	0,020	0,019	0,095	0,091	0,089	0,090
4	1,60	1,90	1,60	1,64	0,047	0,048	0,045	0,041	0,190	0,176	0,179	0,183

Table 4
Values of t_{w_d} (in μsec) obtained experimentally for different block sizes and number of processes. In P4net, HPC160smp and HPC160mc.

With this estimation of the parameters, the model of the execution time (equations 1 and 2) is used to determine the optimum values of the algorithmic parameters for different problem sizes.

In P4net, the best selection is to execute the routine sequentially with small and medium problem sizes, but with problem sizes greater than 4096 it is recommendable to execute the routine in parallel with $p = 2 \times 1$. Table 5 shows the experimental execution time, the theoretical execution time, and the deviation ($\frac{|t_{mod} - t_{exp}|}{t_{exp}}$) between both for different problem sizes, block sizes and logical meshes. The optimum experimental and theoretical times are highlighted. The model makes a good selection of the parameters: number of processors, dimensions of the mesh and block size.

Tables 6 and 7 show the values of the algorithmic parameters provided by our method when following the model (mod.) and the experimental optimum values (opt.). The deviation of the execution time with the parameters provided by the model and the lowest experimental time obtained varying the values of the parameters is also shown. The values of the algorithmic parameters vary for different systems and problem sizes, but with the model and with the inclusion of the cost for different types of MPI communication mechanisms a satisfactory selection of the parameters is made in all the cases. A value 0 in the deviation means the values selected by the model coincide with those with which the lowest execution time was obtained. The selected block size coincides with the optimum in 27 of the 33 cases studied, and the logical topology is selected correctly in 22 of the 28 parallel experiments. In the 10 cases where the selection of the parameters does not give the optimum, the deviation on the execution time is very low. The mean of the deviations is 5.1%.

n	b	logical mesh of processes					
		1×1	1×2	2×1	2×2	1×4	4×1
4096		Experimental time (seconds)					
	32	37,498	31,501	34,673	33,321	27,858	26,068
	64	22,708	21,061	21,337	23,002	22,246	20,913
	128	14,884	17,529	16,279	20,460	21,115	18,293
	256	**13,926**	18,000	16,265	20,840	20,959	19,130
		Theoretical time (seconds)					
	32	38,293	35,668	40,117	36,233	31,497	27,936
	64	22,712	23,480	21,676	23,689	25,354	23,728
	128	14,941	20,262	16,896	21,182	24,382	19,840
	256	**14,233**	20,620	16,992	21,748	25,291	21,327
		Deviation (%)					
	32	2	13	16	9	13	7
	64	0	11	2	3	14	13
	128	0	16	4	4	15	8
	256	2	15	4	4	21	11
5120		Experimental time (seconds)					
	32	70,231	48,747	56,032	49,407	47,625	41,101
	64	43,035	33,511	33,289	35,060	37,187	34,019
	128	28,321	27,059	24,335	29,300	31,366	27,737
	256	25,316	26,342	**23,608**	29,291	29,172	27,822
		Theoretical time (seconds)					
	32	73,024	51,938	63,636	55,254	44,296	43,994
	64	41,073	34,661	33,861	35,969	35,788	36,950
	128	28,076	27,565	25,762	30,997	32,873	30,405
	256	25,306	27,217	**24,712**	31,094	33,604	31,705
		Deviation (%)					
	32	4	7	14	12	7	7
	64	5	3	2	3	4	9
	128	1	2	6	6	5	10
	256	0	3	5	6	15	14

Table 5
Comparison of the theoretical and experimental execution times, for different block sizes and logical meshes of processes, in P4net.

Thus, we can conclude the methodology proposed can be used to obtain execution times close to the optimum without user intervention.

	$p=1$			$p=2$					$p=4$				
	opt.	mod.	dev.	opt.		mod.		dev.	opt.		mod.		dev.
n	b	b	%	b	$r \times c$	b	$r \times c$	%	b	$r \times c$	b	$r \times c$	%
512	64	64	0	64	1×2	128	1×2	2.0	128	1×4	128	1×4	0
1024	128	128	0	128	1×2	64	1×2	3.4	64	4×1	64	1×4	1.2
2048	128	128	0	128	2×1	128	2×1	0	128	1×4	128	2×2	9.6
4096	256	256	0	256	2×1	128	2×1	0.1	128	4×1	128	4×1	0
5120	256	256	0	256	2×1	256	2×1	0	128	4×1	128	4×1	0
Mean			0					1.1					2.2

Table 6
Parameters selection for the Cholesky factorization, in P4net.

	HPC160smp $p=4$					HPC160mc $p=4$					HPC160smp-mc $p=8$				
	opt.		mod.		dev.	opt.		mod.		dev.	opt.		mod.		dev.
n	b	$r \times c$	b	$r \times c$	%	b	$r \times c$	b	$r \times c$	%	b	$r \times c$	b	$r \times c$	%
512	32	4×1	32	4×1	0	32	4×1	32	2×2	81	32	2×4	32	2×4	0
1024	64	4×1	64	4×1	0	64	4×1	32	2×2	62	32	2×4	32	2×4	0
2048	64	4×1	64	4×1	0	64	4×1	32	2×2	1.3	64	2×4	32	2×4	17.2
4096	128	4×1	128	4×1	0	128	2×2	128	4×1	1.6	128	2×4	128	2×4	0
5120	128	4×1	128	4×1	0	128	2×2	128	2×2	0	64	2×4	64	2×4	0
7168	128	4×1	128	4×1	0	128	2×2	128	2×2	0	64	2×4	64	2×4	0
Mean					0					24.3					2.9

Table 7
Parameters selection for the Cholesky factorization in HPC160 with Shared Memory (HPC160smp), MemoryChannel (HPC160mc) and both (HPC160smp-mc).

4. Conclusions

In this paper we have shown that in order to model linear algebra routines to obtain self-optimized routines, it is necessary to use different costs for different types of MPI communication mechanisms (user-defined datatypes or predefined datatypes) and to use different costs for the communication parameters in systems in which it is possible to use more than one interconnection network simultaneously. In these systems it is necessary to decide, in addition to the logical topology of processes, the optimal allocation of processes by node, according to the speed of the interconnection networks. The proposed method has been applied successfully to the Cholesky factorization and may also be applied to other linear algebra routines.

Acknowledgements

This work has been partially supported by Spanish MEC and FEDER under Grant TIC2003-08238-C02-02.

References

[1] Compaq MPI: Description. `http://www.hp.com/techservers/software/cmpidesc.html`.

[2] MPICH-A Portable MPI Implementation. `http://www-unix.mcs.anl.gov/mpi/mpich`.

[3] Zizhong Chen, Jack Dongarra, Piotr Luszczek, and Kenneth Roche. Self-adapting software for numerical linear algebra and LAPACK for clusters. *Parallel Computing*, 29(11-12):1723–1743, 2003.

[4] Jaeyoung Choi, Jack J. Dongarra, L. Susan Ostrouchov, Antoine P. Petitet, David W. Walker, and R. Clint Whaley. Design and implementation of the ScaLAPACK LU, QR, and Cholesky factorization routines. *Sci. Program.*, 5(3):173–184, 1996.

[5] Javier Cuenca, Luis-Pedro García, Domingo Giménez, José González, and Antonio M. Vidal. Empirical modelling of parallel linear algebra routines. In Roman Wyrzykowski, Jack Dongarra, Marcin Paprzycki, and Jerzy Wasniewski, editors, *PPAM*, volume 3019 of *Lecture Notes in Computer Science*, pages 169–174. Springer, 2003.

[6] Javier Cuenca, Domingo Giménez, and José González. Modelling the behaviour of linear algebra algorithms with message-passing. In *Proceedings of the Euromicro Workshop on Parallel and Distributed Processing*, pages 282–289, Mantova, Italy, 2001.

[7] Javier Cuenca, Domingo Giménez, and José González. Architecture of an automatically tuned linear algebra library. *Parallel Comput.*, 30(2):187–210, 2004.

[8] Krister Dackland and Bo Kågström. A hierarchical approach for performance analysis of scalapack-based routines using the distributed linear algebra machine. In *PARA '96: Proceedings of the Third International Workshop on Applied Parallel Computing, Industrial Computation and Optimization*, pages 186–195, London, UK, 1996. Springer-Verlag.

[9] James Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Antoine Petitet, Richard Vuduc, R. Clint Whaley, and Katherine Yelick. Self adapting linear algebra algorithms and software. In *Proceedings of the IEEE: Special Issue on Program Generation, Optimization, and Adaptation*, volume 93, February 2005.

[10] Matteo Frigo and Steven G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume 3, pages 1381–1384, Seattle, WA, may 1998.

[11] Richard B. Gillett. Memory channel network for PCI. *IEEE Micro*, 16(1):12–18, 1996.

[12] Takahiro Katagiri, Kenji Kise, Hiroki Honda, and Toshitsugu Yuba. Effect of auto-tuning with user's knowledge for numerical software. In *CF '04: Proceedings of the 1st conference on Computing frontiers*, pages 12–25, New York, NY, USA, 2004. ACM Press.

[13] Compaq Extended Math Library. `http://h18000.www1.hp.com/math/documentation/cxml/dxml.3dxml.html`.

[14] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI–The Complete Reference: Volume 1, The MPI Core, second edition*. The MIT Press, 1998.

[15] Intel Math Kernel Library v5.2. `http://developer.intel.com/software/products/mkl/mkl52/index.htm`.

[16] R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001.